## Practical Set – 2

**Program 2.1: Implement Fibinacci series using iterative, recursive and generator method. Also compare the performance (time required for execution) of all the three methods..**

```
Import time
# using iterative method
def fibo(n):
   count = 0
   first = 0
   second = 1
   temp = 0
   while count < n:
      print(first)
      temp = first + second
      first = second
      second = temp
      count = count + 1


n = int(input("how long fibonacci series you want ? :"))
start_time=time.time()
fibo(n)
print(time.time()-start_time)
```

**Output :**

```
how long fibonacci series you want ? :21
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
0.003959178924560547
```

**# using recursion**

```
def fibo(n):
   if n <= 1:
      return n
   else:
      return fibo(n-1) + fibo(n-2)

n = int(input("how long fibonacci series you want ? :"))
start_time=time.time()
for i in range(n):
  print(fibo(i))
print(time.time()-start_time)
```

**Output :**

```
how long fibonacci series you want ? :21
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
0.0199735164642334
```

**# using generator method**

```
def fibonacci(n):
   a, b = 0, 1
   for i in range(n):
      yield a
      a , b = b , a+b

n = int(input("how long fibonacci series you want ? :"))
start_time=time.time()
for number in fibonacci(n):
   print (number)
print(time.time()-start_time)
```

**Output:**

```
how long fibonacci series you want ? :21
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
0.002992391586303711
```

**Program 2.2: sort characters of given string. Ex. 'hello' --> 'ehllo'**

```
str = "acknowledgement"
new_str = ''.join(sorted(str))
print("Given String is : "+str)
print("Sorted String is : "+new_str)
```

**Output:**

```
Given String is : acknowledgement
Sorted String is : acdeeegklmnnotw
```

**Program 2.3: Create UDF which find X & Y such that num = XY for given num. Ex. if num = 64, answer should be X=2, Y=6 and X=4,Y=3, and X=8,Y=2**

```
def pair(n):
    global count
    count = 0
    for i in range(1, n + 1):
        rem = n % i
        if rem == 0:
            # print(i)
            for j in range(1, n + 1):
                power = pow(i, j)
                if power == n and j != 1:
```

```
            print("X : ", i)
            print("Y : ", j)
            count = count + 1


n = int(input("Enter the number : "))
pair(n)
if count == 0:
   print("There is not such a pair for given number...")
```

**Output:-**

```
Enter the number : 81
X :  3
Y :  4
X :  9
Y :  2
```

**Program 2.4: Write Generator Function to generate sequence of N random numbers in given range (high, low)**

```
import random
def generate(h,l,n):
   res = random.sample(range(l, h), n)
   yield res


l = int(input("Please Enter lower range : "))
h = int(input("Please Enter higher range : "))
n = int(input("How many numbers you want to generate : "))
m=generate(h,l,n)
print(m.__next__())
```

**Output:-**

```
Please Enter lower range : 11
Please Enter higher range : 23
How many numbers you want to generate : 4
[17, 14, 18, 21]
```

**Program 2.5: check whether the given no is Armstrong or not using recursion.**

```
sum = 0
l = 0

def count(n):
   global l
   if n > 0:
      l = l + 1
      count(n // 10)
   return l
```

```
def armstrong(n, l):
   global sum
   if n > 0:
      rem = n % 10
      sum = sum + (rem ** l)
      armstrong(n // 10, l)
   return sum


n = int(input("Enter the Number : "))

l = count(n)
sum = armstrong(n, l)
if n == sum:
   print("Given number is an armstrong number.")
else:
   print("Given number is not an armstrog number.")
```

**Output:-**

```
Enter the Number : 153
Given number is an armstrong number.
```

### Practical 2.6: Declare a global variable and modify its value using UDF.

```
global glob
glob = 50 # declaration of global variable glob


def change():
   globals()['glob'] = 100    # Modifying value of global variable
   print("Inside the function : ", glob)


print("Real value (Before changing) : ", glob)
change()
print("Outside the function : ", glob)
```

**Output:-**

```
Real value (Before changing) :  50
Inside the function :  100
Outside the function :  100
```